PART I

Visual Study

1 History of Visual Studio

Although this book is dedicated to Visual Studio 2010 and .NET Framework 4.0, having a good historical background in Visual Studio can help you better understand the features treated in the subsequent chapters. Regardless of whether you are old friends with Visual Studio or it is new for you, it is always worth knowing where it started and how it's been evolving.

The roots of Visual Studio go back for almost 19 years, back to the point somewhere between the release of Windows 3.0 and 3.1. It is incredible how the development tool has evolved enormously during almost two decades! The road behind Visual Studio was never smooth or flat; it was full of bumps and curves. However, one thing stayed constant during the years: Microsoft created this tool with developers in mind, and made amazing efforts to build a strong developer community surrounding the product.

In this chapter, you'll read a short story of Visual Studio's past and present, with emphasis on the roots of this great tool, as well as the situations and motivations that led to the integrated development environment (IDE) you use today.

VISUAL STUDIO DOCUMENTARY

At PDC 2009 (held between November 17 and 19, 2009, in Los Angeles), Microsoft published a screencast with the title, "Visual Studio Documentary." This one-hour video is a great source for "company secrets" surrounding Visual Studio from the ancient ages to the present-day stage. A number of Microsoft (and ex-Microsoft) celebrities such as Anders Hejlsberg, Alan Cooper, Bill Gates, Tim Huckaby, Sivaramakichenane Somasegar, Dan Fernandez, Tony Goodhew, Jason Zander, Scott Guthrie, and Steve Balmer are featured in this video. They add interesting personal commentaries on history, motivations, technology context, competitors, and nitty-gritties that paved the road for Visual Studio.

You can download this two-part documentary from `http://channel9.msdn.com/shows/VisualStudioDocumentary/The-Visual-Studio-Documentary-Part-One`, where you can also find the link for the second part of the video.

ROOTS

For a long time, Windows development was a field where only C and C++ programmers could play. They had to carry out a lot of tasks for creating the simplest user interface — such as defining and registering Windows classes, implementing the Windows message loop, dispatching Windows messages, painting the client in Windows, and so on. The smallest "Hello, World" program for Windows was about a hundred lines of code, where you could not meet any explicit statement to print out the "Hello, World" text. Instead, you had to draw this text to an absolute window position as a response to the `WM_PAINT` message. At that time, the user interface (UI) was defined by static text files that were compiled into binary resources and linked to the application. The UI missed the concept of controls — there were windows and child windows, all of them represented by HWNDs (or window handles).

At that time, developers accepted this way of Windows software creation as a price for interacting with a graphical user interface (GUI).

The First Breakthrough: Visual Basic

The first tool that dramatically changed Windows application development was Visual Basic 1.0, released in May 1991. Visual Basic introduced (or, perhaps, invented) such concepts as forms, controls, code-behind files — all of which are still in use in contemporary development tools. Instead of writing resource files and addressing UI elements through 16-bit constants, you could drag-and-drop predefined UI controls to your forms and program their events. The hundred-line "Hello, World" program was so simple with Visual Basic:

```
Private Sub Form_Load()
    MsgBox("Hello, World!")
End Sub
```

You did not have to care about programming the message loop or event dispatching code! Visual Basic allowed you to create an application represented by an icon on the Windows desktop. When you double-clicked on that icon, the application started and ran just as Word or Excel — which, at that time, was a delightful experience. Visual Basic revolutionized the application development platform, because it made Windows programming available for the masses.

Other Languages and Tools

The whole visual composition aspect of Visual Basic was something that could be applied for the C++ and other languages as well. In the few years following the release of Visual Basic, a plethora of tools was created by Microsoft:

- Visual C++ 1.0 was released in February 1993 with Microsoft Foundation Classes (MFC) 2.0 and proved that C++ programming for Windows could be more productive than ever before — while still keeping the full and granular control over the operating system.
- In 1992, Fox Technologies (the creator of FoxBASE and FoxPro) merged with Microsoft, and, at the end of 1995, Visual FoxPro 3.0 was released.
- The emergence of the Java programming language in 1995 motivated Microsoft to create its own Java language implementation. It was Visual J++1.0 that conformed to the Sun specification and used Microsoft's Java Virtual Machine (JVM).

Having so many separate languages and tools, the architect teams recognized that the whole visual aspect could be separated from the languages. Why create separate IDEs for all the languages and tools if they could fit into the same environment? That was when the idea of Visual Studio was born.

Visual Studio 97 and 6.0

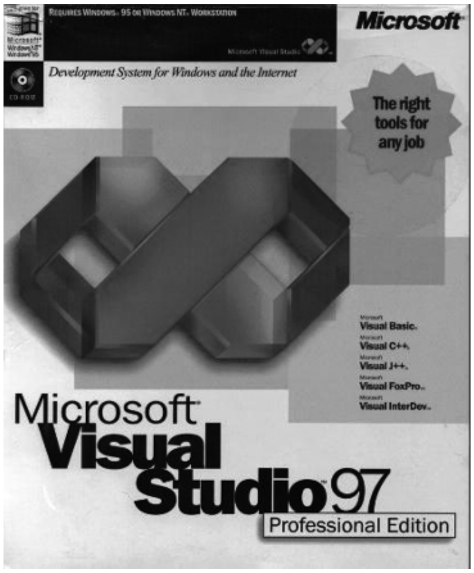
In 1997, Microsoft built a single environment to integrate multiple languages into one application surface. This was released as Visual Studio 97, bundling Microsoft development tools for the first time. This package contained Visual Basic 5.0, Visual C++ 5.0, Visual FoxPro 5.0, and Visual J++ 1.1 from the set of existing tools. The bundle was also extended with Visual InterDev, a new tool for developing dynamically generated Web sites using the Active Server Pages (ASP) technology. A snapshot of the Microsoft Developer Network Library was also a part of the package.

At this time, the IDE named Developer Studio integrated only Visual C++, J++, Visual InterDev, and MSDN. The name “Visual Studio” was rather the name of the bundle (because Visual Basic and Visual FoxPro had their own IDEs).

The famous and long-lived logo of Visual Studio that resembles the sign of infinity (or to the Moebius strip) was introduced with the first version. You can clearly recognize it from the package cover shown in Figure 1-1.

Shortly after the 1997 version, in June 1998, Visual Studio 6.0 was released. It did not contain too many new things, but fixed early integration issues to make the product more robust. The version numbers of all of its constituent parts also moved to 6.0 to suggest a higher level of integrity among the individual tools. However, instead of three IDEs in Visual Studio 97, version 6.0 had four, because Visual C++ got its own IDE.

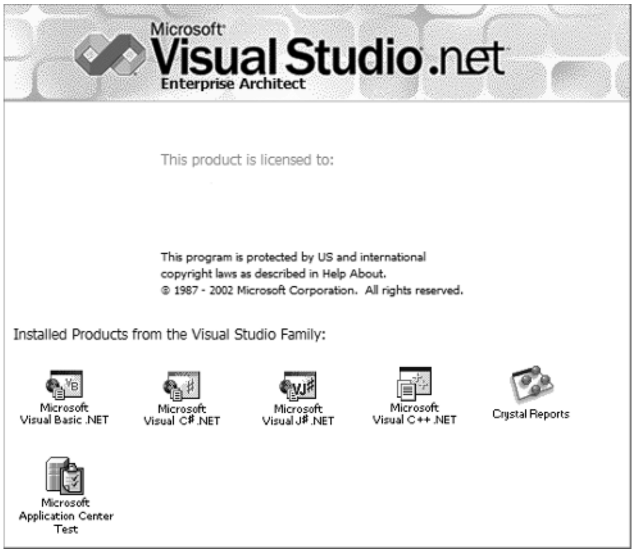
Microsoft understood the challenge of the Java phenomenon. Not only the language, but also the managed nature of the Java platform inspired the company to make a huge leap in regard to a development platform shift. The huge amount of research and development work done between 1998 and 2002 led to the introduction of the .NET Framework. This new platform entirely changed the future of Visual Studio.


Microsoft
Development System for Windows and the Internet
The right tools for any job
Visual Basic
Visual C++
Visual J++
Visual FoxPro
Visual InterDev
Microsoft Visual Studio 97
Professional Edition
FIGURE 1-1: The Visual Studio 97 package

VISUAL STUDIO.NET 2002 AND 2003

In July 2000, the .NET Framework was first announced publicly at Professional Developers Conference (PDC) in Orlando, Florida. At PDC, Microsoft also demonstrated C#, and announced ASP+ (which was later renamed to ASP.NET) and Visual Studio.NET. It took more than a year and a half, but, in February 2002, .NET Framework 1.0 was released as part of a pair with Visual Studio.NET (the latter of which is often referred as Visual Studio .NET 2002).

Visual Studio.NET had an IDE that finally integrated the tools and languages into the same environment. Because (except for Visual C++) all the languages were new (even Visual Basic .NET could be considered as new, because it had been fundamentally changed), the toolset had to be re-designed and re-implemented. Microsoft had a better chance to ultimately integrate the pieces into a single IDE, and it did so remarkably. Figure 1-2 shows the splash screen of Visual Studio.NET Enterprise Architect Edition, which indicates that constituent languages and tools share a common IDE.



This product is licensed to:

This program is protected by US and international copyright laws as described in Help About.
© 1987 - 2002 Microsoft Corporation. All rights reserved.

Installed Products from the Visual Studio Family:

FIGURE 1-2: Visual Studio.NET splash screen

The set of languages Microsoft integrated into the product were established with long-term support for the .NET Framework in mind. At that time, developers could use four languages out-of-the-box:

- *Visual C#* — This completely new language was developed (by a team led by Anders Hejlsberg) and enormously used by Microsoft itself to develop the Base Class Library of the framework. This new language attracted a lot of developers both from the former Visual Basic and C++ camps, and became very popular. It uses C-like syntax ("curly-braced-language"), but its constructs are more readable than those of C or C++.
- *Visual Basic .NET* — The former Visual Basic versions just scratched the surface of object-oriented programming (OOP), but the real innovations were missing from the language for a long time. The clear object-oriented nature of .NET required a new Visual Basic. Microsoft recognized the popularity of the language and created Visual Basic .NET with full .NET and OOP support.

- *Visual C++* — With the ascension of .NET, there were still many software development areas with native (Win32 API) Windows development rules (for example, device driver implementation). Visual C++ provided this capability. Besides, Visual C++ was able to interoperate with managed code, and additional grammatical and syntactic extensions (Managed Extensions for C++) allowed compiling code targeting the .NET Common Language Run-time (CLR).
- *Visual J#* — This language was considered as a replacement for Visual J++. However, this language had a Java syntax. It could build applications targeting only the .NET Framework's CLR. Now having a competing platform against Java, after replacing J++, Microsoft no longer created any language running on the JVM.

The .NET Framework's Base Class Library was established as a common infrastructure for developers, thus making it easy and natural to solve common tasks such as using data access and Web services. Visual Studio .NET provided a rich set of built-in tools to leverage the infrastructure provided by the framework. The IDE was designed with extensibility in mind, and allowed developers to integrate their own custom tools into the IDE.

A bit more than a year after Visual Studio.NET was released, a new version, Visual Studio .NET 2003, was shipped together with .NET Framework 1.1. Microsoft had a lot of work to do to stabilize the framework, and, of course, dozens of critical bugs were fixed. A few things (such as the security model) were also changed, and new features were added to the framework (such as built-in support for building mobile applications, IPv6 support, and built-in data access for ODBC and Oracle databases). Also, the CLR became more stable from version 1.0 to 1.1.

Visual Studio.NET (the one released with .NET 1.0) was not able to compile applications for the new CLR version, so the 2003 version had to undertake this task. Thanks to the robustness and stability of Visual Studio .NET 2003, it became very popular, and is still in use because of the large number of business applications developed for .NET 1.1.

VISUAL STUDIO 2005

Released in November 2005, Visual Studio 2005, together with .NET Framework 2.0, brought fundamental changes to the tool, as well as to the languages. The Common Type System (CTS) of the framework introduced generic types. This concept affected all languages, because they must have been prepared to handle the feature of generics, and development tools also needed to encapsulate support for this. The shift of CTS also touched ASP.NET and ADO.NET.

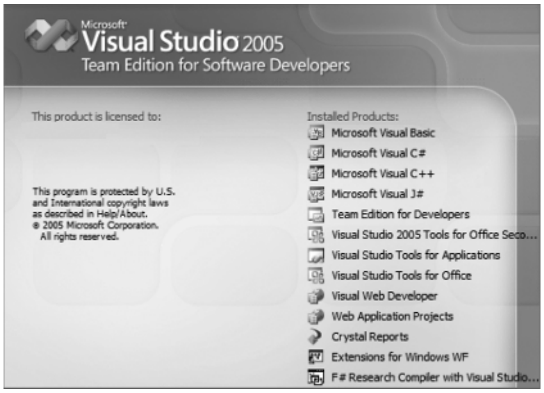
Web application development had some pain in the former Visual Studio versions. Developers had to install and use Internet Information Server (IIS) locally on their machines, and it meant confrontation with system administrators who did not want to have IIS on desktops for security reasons. Visual Studio 2005 installed a local development Web server on desktops and resolved this particular situation.

With this release, Microsoft widened the camp of programmers using Visual Studio with two new editions:

- *Express Editions* — These editions (they are free) targeted students, hobbyists, and other developers coding for fun. Instead of giving a "geese" version of Visual Studio for free, Microsoft created language-related kits with the names of Visual C# 2005 Express, Visual Basic 2005 Express, Visual Web Developer, and Visual C++ 2005 Express, equipped with the full language feature set, but with limited tool support.
- *Team System Editions* — Microsoft wanted to move Visual Studio out of the box of development tools and position it among the high-end enterprise development tools. Team System Editions provided out-of-the-box integration with Microsoft's Team Foundation Server 2005, and added powerful productivity tools for specific development project roles. There are four editions for Developers, Testers, Architects, Database Designers, and a fifth one, Visual Studio Team Suite, which includes all of the features of these four editions in a single package.

Compare the list of installed products in the splash screen of Visual Studio 2005 Team Edition for Software Developers (shown in Figure 1-3) with the list shown in Figure 1-2. The eye-catching difference tells you how many tools were added to the new editions.

Following the initial release, a few special-purpose products were also shipped and integrated into the IDE (such as Visual Studio Tools for Office and Visual Studio Tools for Applications).


Microsoft Visual Studio 2005
Team Edition for Software Developers

This product is licensed to:

This program is protected by U.S. and International copyright laws as described in Help/About.
© 2005 Microsoft Corporation.
All rights reserved.

Installed Products:
Microsoft Visual Basic
Microsoft Visual C#
Microsoft Visual C++
Microsoft Visual J#
Team Edition for Developers
Visual Studio 2005 Tools for Office Seco...
Visual Studio Tools for Applications
Visual Studio Tools for Office
Visual Web Developer
Web Application Projects
Crystal Reports
Extensions for Windows WF
F# Research Compiler with Visual Studio...

FIGURE 1-3: Products installed with Visual Studio 2005 Team Edition for Software Developers

An unusual thing happened in November 2006: .NET Framework 3.0 was released without any accompanying Visual Studio version. This major .NET version kept the CLR untouched and added infrastructure components to the framework — Windows Workflow Foundations (WF), Windows Communication Foundations (WCF), Windows Presentation Foundation (WPF), and CardSpace. Developers could download Visual Studio extensions to use these new .NET 3.0 technologies.

VISUAL STUDIO 2008

In November 2007, one year after .NET 3.0, Visual Studio 2008 was shipped together with .NET Framework 3.5. Although the .NET CLR was still version 2.0, the new query expression syntax (LINQ) feature in .NET 3.5 demanded changes to the existing tools.

The most popular feature of version 2008 was multi-targeting. With this, Visual Studio developers could specify the target framework (.NET 2.0, .NET 3.0, and .NET 3.5) of their projects, or even mix projects with different targets in their solutions. Because one native Win32 process could host only one CLR at the same time, .NET 1.1 (because it uses CLR 1.1) was not in the list of available targets.

Both Visual Basic and C# went through fundamental changes to support the new LINQ syntax. As an addition, Visual Basic 9.0 was given support for XML literals (including plain XML text in the source code); C# 3.0 was extended with new initializer syntax. Both languages were equipped with new constructs (including type inference, anonymous types, extension methods, and lambda expressions) to support LINQ and reduce syntax noise.

The J# language was retired in Visual Studio 2008; the last version supporting it was Visual Studio 2005. Microsoft made this decision because the use of J# started to decline. However, the last version of J# will be supported until 2015.

The LINQ technology was about moving data access and data processing toward the functional programming paradigm. This new paradigm (new for Microsoft development tools) gained momentum as Microsoft Research started to work on a new functional programming language called F#. The first community technology preview (CTP) of the language appeared in Visual Studio 2005 (take a look again at the last product item in Figure 1-3), and Visual Studio 2008 hosted a few more new CTPs.

In addition to the main themes of .NET Framework 3.5, Visual Studio has other great features and changes:

- Built-in support for the three foundations released in .NET 3.0 and refreshed in 3.5:
    - WPF has a visual designer for XAML layouts.
    - WCF has a few project types out-of-the-box.
    - WF has visual a designer to create workflows graphically.
- JavaScript programming is now supported with IntelliSense and a debugger.
- Web developers can use a new and powerful XHTML/CSS editor.

After the initial release, Microsoft's new technologies were also integrated with Visual Studio:

- One of the new emerging technologies was Silverlight. With the initial Visual Studio release in November 2007, only Silverlight 1.0 was available, and that was based on JavaScript. In August 2008, Silverlight 2.0 was shipped, implementing the same full CLR version as .NET Framework 3.0, and so it could execute programs written in any .NET language. In July 2009, Silverlight 3.0 was released. All versions had their own toolset that can be downloaded and integrated with Visual Studio 2008.
- In August 2008, a service release was issued with .NET Framework 3.5 SP1 and Visual Studio 2008 SP1. This version added new ADO.NET data features to the framework and also designers to the IDE:
    - *ADO.NET Entity Framework* — This raises the level of abstraction at which programmers work with data to the conceptual level.

- *ADO.NET Data Services* — This is first-class infrastructure for developing dynamic Internet components by enabling data to be exposed as REST-based data services.
- *ASP.NET Dynamic Data* — This provides a rich scaffolding framework that allows rapid data driven development without writing any code.

Visual Studio 2008 did not change the structure of editions in version 2005. All editions (including Visual Studio Team System 2008 and Visual Studio 2008 Express Editions) were released together.

VISUAL STUDIO 2010

The latest version of Visual Studio has 10.0 as the internal version, and its name is officially Visual Studio 2010.

No doubt, Microsoft takes Visual Studio into account as the ultimate tool for developers creating applications and business solutions on the Windows platform. This intention can be caught on the messages called "the pillars of Visual Studio 2010":

- *Creativity Unleashed* — You can use prototyping, modeling, and visual design tools to create solid, modern, and visionary solutions through software development. You can leverage the creative strengths of your team to build your imaginations together.
- *Simplicity through Integration* — Visual Studio helps simplifying common tasks, and helps you explore the depth of the platform you and your team work with. It has an integrated environment, where all team members can use their existing skills to model, code, debug, test, and deploy a growing number of application types, including the solutions for the cloud platform.
- *Quality Code Ensured* — The toolset of Visual Studio includes everything that helps you with maintaining source code, finding and fixing bugs, and managing your projects. Testers and developers on your team can use manual and automated testing, as well as advanced debugging tools, from the very beginning. Utilizing these tools, you can be confident that the right application is built, the right way.

These messages are a very brief and straightforward summary of what Visual Studio 2010 offers for experts — software developers, testers, architects, business analysts, project managers — working on software development tasks and projects.

Changes in Editions

While Visual Studio 2008 had many editions — such as Standard, Professional, and Team System Editions (including Development, Database, Architecture, and Test Editions) — you will be able to choose from three main version (of course, free Express editions are still available):

➤ *Microsoft Visual Studio 2010 Professional with MSDN* — This version is intended to be the essential tool for basic development tasks to assist developers in easily implementing their ideas.

➤ *Microsoft Visual Studio 2010 Premium with MSDN* — This provides a complete toolset to help developers deliver scalable, high-quality applications.

➤ *Microsoft Visual Studio 2010 Ultimate with MSDN* — This version (as its name suggests) is a comprehensive suite of application life-cycle management tools for software teams to help ensure quality results from design to deployment.

The feature sets of these editions are formed so that editions contain every feature the lower editions have, plus add their own functionality on top of them.

Microsoft's intention with Visual Studio 2010 is clear from the features all editions have in common:

➤ *Development platform support* — All important platforms (Windows, Web, Office, SharePoint, and cloud development) are available with a common tool set.

➤ *Team Foundation Server integration* — There is no difference among the Visual Studio editions in the Team Foundation Server support they have! All of them are shipped with the Visual Studio Team Explorer 2010 to instantly access Team Foundation Server with the entire feature set, including source control and work item management, build automation and test case management, Team Portal, Business Intelligence (BI), and reporting.

➤ *Debugging and Diagnostics* — The efficient debugging and diagnostics tools (with a number of new Visual Studio 2010 innovations) help developers to become more productive than ever before. Now, post-mortem debugging, multi-threaded application debugging through the Parallel Stack and Tasks window, and 64-bit support for mixed mode debugging are available for every developer independently of the edition he or she uses.

These editions are bundled with MSDN subscriptions. This is a great benefit — especially for Premium and Ultimate users who receive additional software for production use (such as Expression Studio 3, Office Plus 2010, Visio Premium 2010, and Project Professional 2010). All users get the standard MSDN subscription benefits, such as priority support on MSDN Forums, technical support incidents, *MSDN* magazine, and so on.

As a result of setting up the editions as treated, small developer teams with the Professional edition now can work together in a way that was possible only with one of the Team System editions with the previous versions. The Premium edition adds new tools for database development, testing, and advanced functions for debugging and diagnostics. Users of the Ultimate edition have architecture, modeling, and test lab management tools shipped with the product, a benefit they never got before with Visual Studio.

What's New in Visual Studio 2010

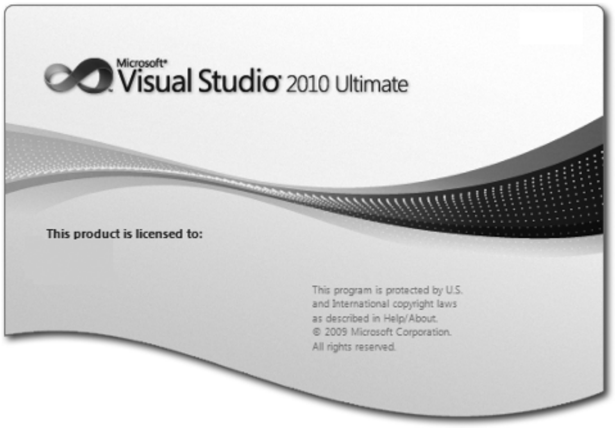
Addressing what is new in Visual Studio is not tackled here in its entirety. Each chapter of this book contains sections dedicated to this topic. Moreover, many chapters are especially about treating Visual Studio new features with all nitty-gritty details.

Without the need of completeness, here is a short list to whet your appetite:

- Cloud development (Windows Azure) and SharePoint development is now supported.
- Test Driven Development (TDD) is available in Visual Studio. You can follow the Consume-First-Declare-Later approach during code writing.
- The code editing experience has been significantly enhanced:
    - Visual Studio now understands your code, provides you with Call Hierarchy, and highlights references.
    - With the Quick Search function, you can easily navigate within your code — not just in the current code file but in the entire solution.
    - IntelliSense has been improved. It now has substring matching, helping you when you do not remember exact member names.
    - The new code editor is extensible, and creating extensions has been significantly simplified.
- Online Visual Studio Gallery is integrated directly into Visual Studio. With the Extension Manager, you can browse online content (tools, controls, and templates) and immediately install third-party extensions.
- You are not obliged to create new projects from the templates already installed on your machine. You can create your project right from online project templates with the New Project dialog.
- Multi-core and multi-threaded applications are now first-class citizens in Visual Studio. You can debug your applications with their nature of using multiple parallel tasks and threads. The new tools and views allow you to look for and focus on those details (race conditions, blockings, interoperation, and so on) that were invisible in previous versions.
- Modeling, designing, and validating architecture now are organic parts of Visual Studio. Not only can architects benefit from these features, but those can be used for communication among team members or with customers.

Shift to WPF

Maybe it sounds weird, but the majority of Visual Studio's code base is unmanaged code — large pieces of this code come from the COM era, and did not really change over time. With Visual Studio 2010, the development team undertook the challenge of a technology shift: the UI technology of the shell and a major part of the IDE was changed from GDI/GDI+ to WPF — that is, a managed technology. The new design of the product (the new splash screen is shown in Figure 1-4) communicates this new approach.



This product is licensed to:

This program is protected by U.S. and International copyright laws as described in Help/About. © 2009 Microsoft Corporation. All rights reserved.

FIGURE 1-4: The splash screen reflects the brand new design of Visual Studio 2010

The formerly angular and multi-colored "infinity sign" logo became a round-cornered and gradually-colored one, emphasizing the smooth integration among the tools within the IDE.

The new code editor of Visual Studio has been totally rewritten to use WPF as its display technology, new functions such as the modeling tools, the new parallel task debugger, and many, many more features also were implemented with WPF.

SUMMARY

The name of Visual Studio is about 13 years old, but the roots of the product go back almost two decades. The first milestone on the road was definitely Visual Basic 1.0. Up until 1997, other programming languages and tools also picked up the visual composition aspect that had distinguished Visual Basic from the formerly used development tools. Visual Studio was born by packaging these tools (Visual Basic, Visual C++, Visual FoxPro, Visual J++, Visual InterDev, and MSDN) into a bundle.

With the introduction of the .NET Framework in 2002, Visual Studio started to gain big momentum. The languages and tools encapsulated into the product changed, together with the state-of-the-art paradigms and trends. The surrounding developer community has experienced a spectacular growth in the last eight years.

After seven major versions (five of which leveraged the .NET Framework), Visual Studio transformed from a single development tool into a rock-solid software development environment. It supports a wide audience, from students and hobbyists, to large-enterprise IT groups with full application life-cycle management functionality needs.

As of this writing, you can use Visual Studio 2010 released together with .NET Framework 4.0. Its functionality has been significantly extended since Visual Studio 2008.

In Chapter 2, you'll learn about the new enhancements of the IDE's user interface.